

**FAKULTÄT INFORMATIK**

Masterstudiengang Informatik

**Entwicklerdokumentation zur Anpassung des ADAMO Modelers**

Im Rahmen eines Master-Studienprojekts

Masood Ahmed

Jennifer Espich

Christina Frank

Granit Gecaj

Daniel Lackmann

Markus Schmidtner

Betreuer:

Daniel Hilpoltsteiner

Prof. Dr. rer. oec. Christian Seel

**Inhaltsverzeichnis**

[1. Allgemeine Beschreibung 3](#_Toc521236170)

[2. Technischer Aufbau 4](#_Toc521236171)

[2.1 Datenbank 4](#_Toc521236172)

[2.2 MQTT 4](#_Toc521236173)

[3. Installationsanleitung 5](#_Toc521236174)

[3.1 GitKraken 5](#_Toc521236175)

[3.2 Visual Studio Code 6](#_Toc521236176)

[3.3 express.js 6](#_Toc521236177)

[3.4 Postgres 7](#_Toc521236178)

[3.5 pgAdmin 7](#_Toc521236179)

[3.5 OpenVPN 7](#_Toc521236180)

[4. Lokale und Server-Datenbank 9](#_Toc521236181)

[4.1 Anlage einer lokalen Datenbank 9](#_Toc521236182)

[4.2 Wechsel zwischen lokaler und Server-Datenbank 9](#_Toc521236183)

[5. Bedienungsanleitung 9](#_Toc521236184)

[**A Literaturverzeichnis** I](#_Toc521236185)

[B Abbildungsverzeichnis II](#_Toc521236186)

[**1** Literaturverzeichnis II](#_Toc521236187)

1. Allgemeine Beschreibung

Camunda stellt einen Modeler bereit, der sowohl für die Prozess- als auch die Entscheidungsmodellierung geeignet ist und die Modellierungssprachen BPMN 2.0, CMMN 1.0 und DMN 1.1 unterstützt.[[1]](#footnote-1) Aufgrund der eingeschränkten Funktionalität und neuer Bedürfnisse von Anwendern wird dieses Werkzeug in Zusammenarbeit des Instituts für Projektmanagement und Informationsmodellierung (IPIM) und dem Kompetenznetzwerk Intelligente Produktionslogistik (EFRE) zum adaptiven Modeler (ADAMO) erweitert.

Als neue Funktion zusätzlich derer von Camunda befähigt der Modeler auf BPMN basierende Prozesse ortsunabhängig, simultan und kollaborativ zu visualisieren. Hierfür wird ein Publisher/Subscriber Modell mit Hilfe des Protokolls MQTT verwendet.

Eine weitere Anwendung ist das Auswerten von P

Auswerten von Subprozessen

Auswerten von Variablen

1. Technischer Aufbau

2.1 Datenbank

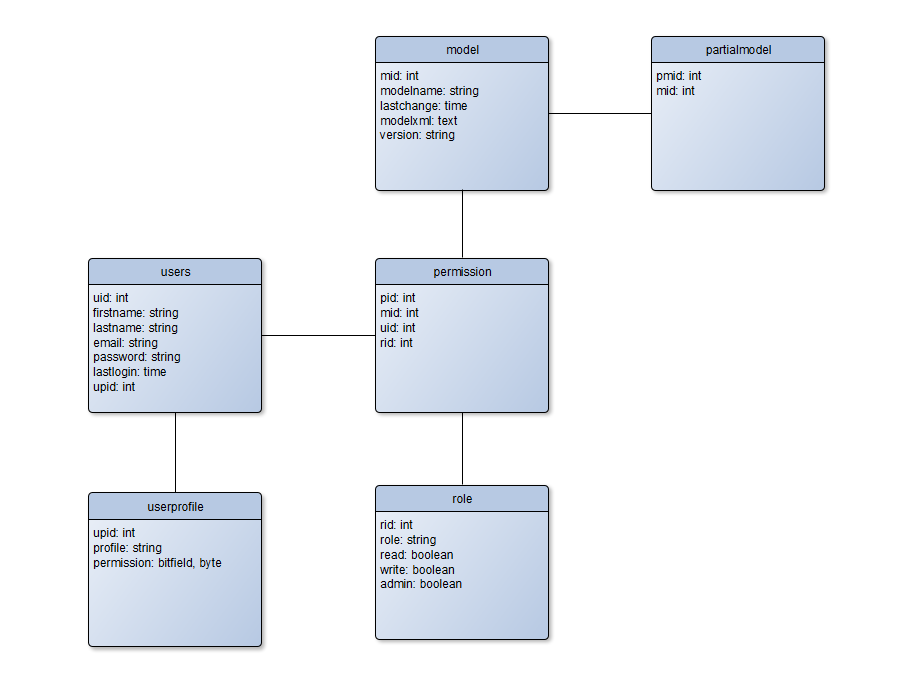


Abbildung 1: Datenbanktabellen und ihre Beziehungen zueinander

2.2 MQTT

2.3 Express.js

1. Installationsanleitung

3.1 GitKraken

Zur Versionierung von Code-Änderungen wird ein Git-Repository verwendet. Um dieses benutzerfreundlich zu verwalten kann GitKraken verwendet werden, zu dem ein Download unter <https://www.gitkraken.com/> möglich ist.

Im nächsten Schritt muss das Repository geklont werden. Dieses befindet sich auf GitHub unter <https://github.com/HAWMobileSystems/IntSys>, wofür zum Klonen folgender Link verwendet wird: [git@github.com:HAWMobileSystems/In tSys.git](mailto:git@github.com:HAWMobileSystems/In%20tSys.git).

Dafür GitKraken nach erfolgreichem Download und Installation starten und genannte URL unter

„File“ -> „Clone Repo“

einfügen. Im nachfolgenden Beispiel sollte das Ziel durch einen lokal sinnvollen, leicht aufzufindenden Pfad ersetzt werden.

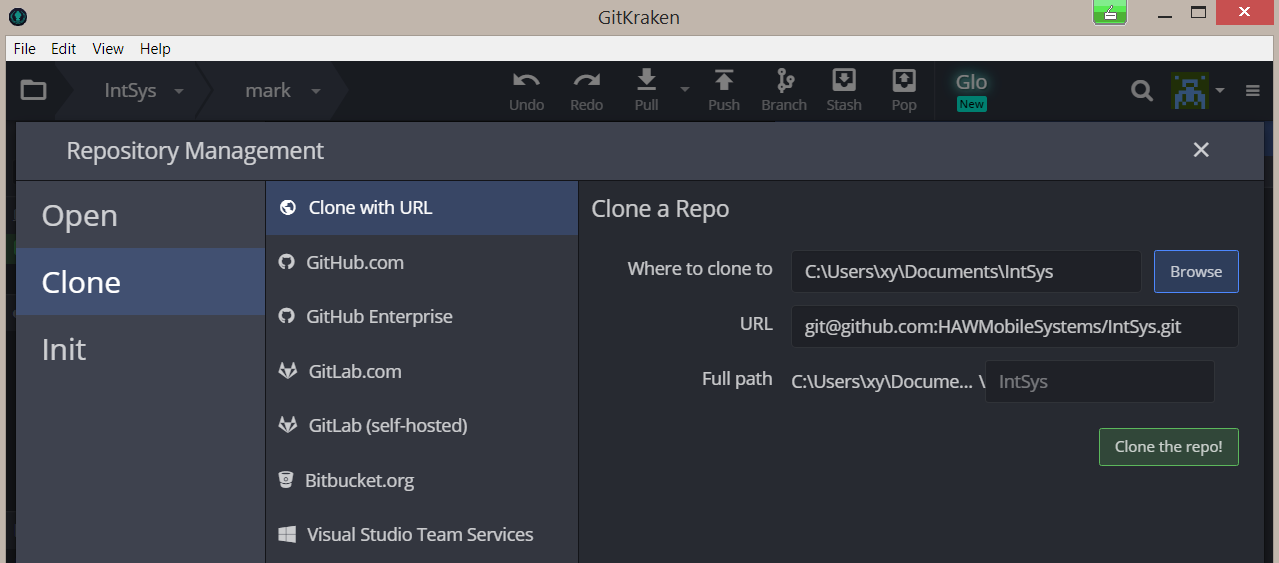


Abbildung 2: Klonen eines Repositories mittels GitKraken

3.2 Visual Studio Code

Außerdem wird für weitere Implementierungen eine Entwicklungsumgebung benötigt, wofür Visual Studio Code eingesetzt werden kann, da dies ebenso Git unterstützt. Ein weiterer Vorteil dieses Tools ist das leichte Debuggen von Webanwendungen. Der Installer hierfür ist unter <https://code.visualstudio.com> zu finden.

Als zusätzliche Komponenten werden vom Hersteller selbst Git, Node.js und TypeScript empfohlen, dies ist jedoch mit den dependencies im Projekt bzw. dem im Nachgang erläuterten express.js bereits abgedeckt. Als empfehlenswerte Erweiterungen bezüglich Code-Style zeigen sich jedoch die \*lint Komponenten. Diese können wie im nachfolgenden Screenshot gezeigt mittels des gelb markierten Buttons gefiltert, selektiert und installiert werden.

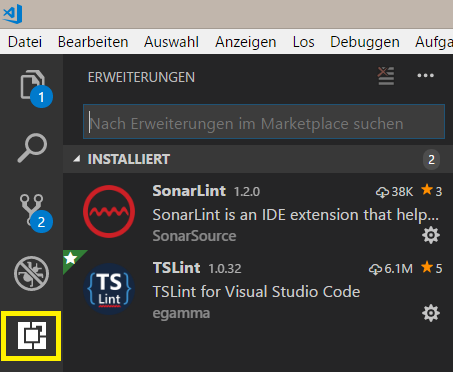


Abbildung : Installation von zusätzlichen Komponenten

3.3 express.js

Für die Realisierung des Servers und weiterer auf JavaScript basierender Webentwicklungen durchzuführen ist node.js ein hilfreiches Framework. Beim Modeler ist die Erweiterung express.js im Einsatz. Der Download steht unter <http://expressjs.com/de/> zur Verfügung, wobei die bereitgestellte Installationsanleitung verwendet werden kann: <http://expressjs.com/de/starter/installing.html>. Die Installation kann im Terminal von Visual Studio Code durchgeführt werden.

TODO: Schritte im Terminal erklären

3.4 Postgres

Optional, wird nur benötigt falls eine lokale Datenbank genutzt werden soll.

Ein Download von Postgres ist unter <https://www.postgresql.org/download/> möglich.

3.5 pgAdmin

TODO

3.5 OpenVPN

Da sich die Datenbank auf einem Server im Hochschulnetz befindet, wird eine VPN-Verbindung benötigt.

Die Hochschule stellt hierfür eine Open VPN GUI bereit, welcher unter folgendem Link für das gewünschte Betriebssystem heruntergeladen werden kann:

<https://www.haw-landshut.de/hochschule/zentrale-services/service-it/download.html>

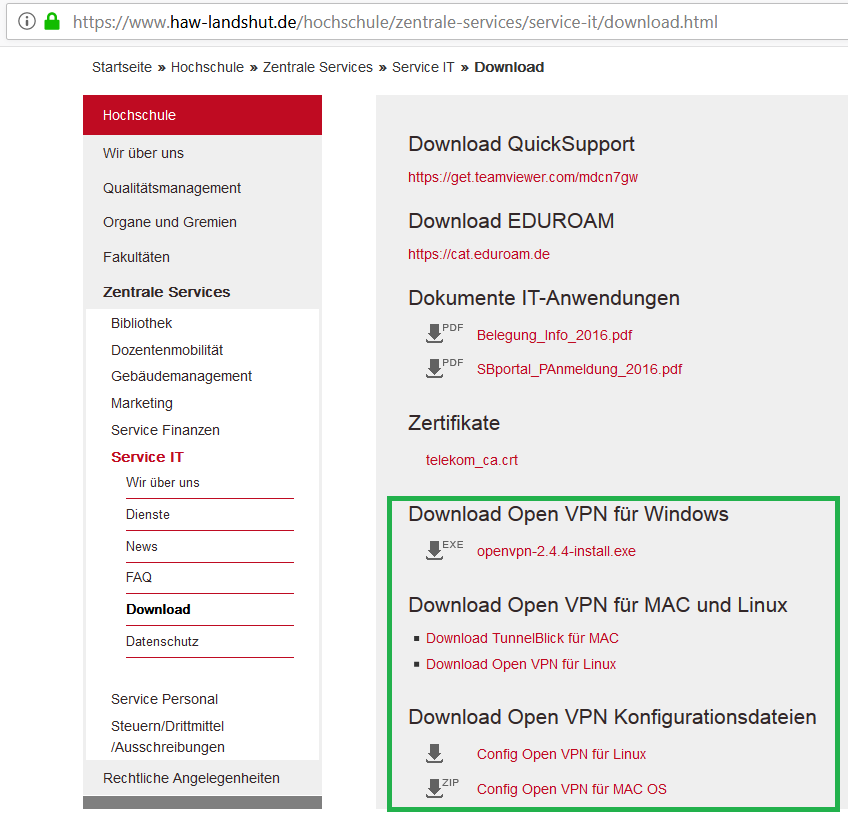


Abbildung 4: Download-Möglichkeiten von Open VPN von der Homepage der HAW Landshut

1. Lokale und Server-Datenbank

4.1 Anlage einer lokalen Datenbank

TODO: Aktuelle Creates einfügen

4.2 Wechsel zwischen lokaler und Server-Datenbank

1. Bedienungsanleitung für Entwickler

Nachfolgend beschriebene Schritte müssen bei jedem Start des Projekts durchgeführt werden.

* Entweder die lokale Datenbank aktiv haben (siehe 4.2) oder mit VPN zur Server-Datenbank verbinden
* Visual Studio Code starten
  + Falls das Terminal nicht automatisch geöffnet ist, dieses über den Menüpunkt „Anzeigen“ -> „Integriertes Terminal“ öffnen
  + Mit cd API in den API Ordner wechseln
  + node index.js ausführen, um startup, routing und andere Basisfunktionen zu starten. Falls eine Fehlermeldung bezüglich Timeout erscheint, sollte die VPN-Verbindung geprüft werden

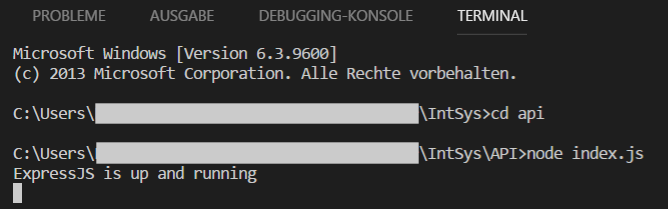


Abbildung 5: index.js starten

* + Um die Kommunikation zwischen mehreren Endgeräten zu testen, muss der MQTT gestartet werden. Hierzu eine neue Konsole öffnen und mit  
    cd API abermals in den API-Ordner wechseln
  + node mqttserver.js ausführen, um den MQTT starten.

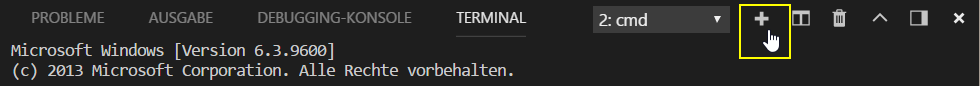


Abbildung 6: Öffnen einer neuen Konsole

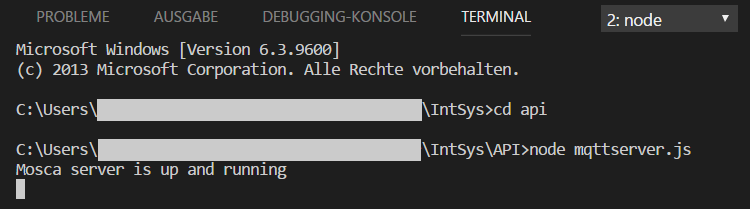


Abbildung 7: Starten des MQTT-Servers

* Eine dritte Konsole öffnen und mit cd angular2 ins Angular-Verzeichnis wechseln
* Den express.js bzw. node.js Paketmanager mit npm start starten.   
  Im Idealfall erscheinen Erfolgsmeldungen, andernfalls werden bei einem nicht complierbarem Code hier die Fehlerstellen angezeigt. Wenn trotz genauerer Betrachtung des Codings keine Fehlerquellen auffindbar sind, hilft ein erneutes npm install. Dieses ist beispielsweise nach Änderungen an den dependencies notwendig.

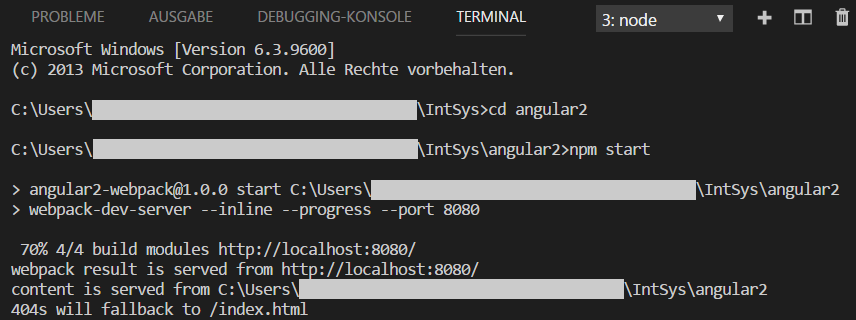


Abbildung 8: Starten des Node-Paketmanagers

* Im Browser kann die Oberfläche unter localhost:8080 erreicht werden
* Login-Daten für Testzwecke:

User: 1

Passwort: 12341234

* Über das „+“-Symbol kann ein neues Modell geöffnet werden

**A Literaturverzeichnis**

Camunda Services GmbH (Hg.) (2018): Modeler. Online verfügbar unter https://camunda.com/products/modeler/, zuletzt aktualisiert am 02.08.2018, zuletzt geprüft am 03.08.2018.

B Abbildungsverzeichnis

[Abbildung 1: Datenbanktabellen und ihre Beziehungen zueinander 4](#_Toc521236205)

[Abbildung 2: Klonen eines Repositories mittels GitKraken 5](#_Toc521236206)

[Abbildung 3: Installation von zusätzlichen Komponenten 6](#_Toc521236207)

[Abbildung 4: Download-Möglichkeiten von Open VPN von der Homepage der HAW Landshut 8](#_Toc521236208)

[Abbildung 5: index.js starten 10](#_Toc521236209)

[Abbildung 6: Öffnen einer neuen Konsole 10](#_Toc521236210)

[Abbildung 7: Starten des MQTT-Servers 10](#_Toc521236211)

[Abbildung 8: Starten des Node-Paketmanagers 11](#_Toc521236212)

1. Camunda Services GmbH 2018. [↑](#footnote-ref-1)